DTIC FILE COPY

4

AD-A197 772

AD

TECHNICAL REPORT ARCCB-TR-88017

AN ADAPTIVE METHOD WITH MESH MOVING AND LOCAL MESH REFINEMENT FOR TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

DAVID C. ARNEY

JOSEPH E. FLAHERTY

APRIL 1988

US ARMY ARMAMENT RESEARCH,
DEVELOPMENT AND ENGINEERING CENTER
CLOSE COMBAT ARMAMENTS CENTER
BENÉT LABORATORIES
WATERVLIET, N.Y. 12189-4050

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARCCB-TR-88017	2. GOVT ACCESSION NO. A197 772	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN ADAPTIVE METHOD WITH MESH MOVING AND LOCAL MESH REFINEMENT FOR TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David C. Arney and Joseph E. Flaherty (See Reverse)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army ARDEC Benet Laboratories, SMCAR-CCB-TL Watervliet, NY 12189-4050		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS AMCMS No. 6111.02.H600.0 PRON No. 1A6DZ602NMSC
11. CONTROLLING OFFICE NAME AND ADDRESS US Army ARDEC Close Combat Armaments Center Picatinny Arsenal, NJ 07806-5000		12. REPORT DATE April 1988
		13. NUMBER OF PAGES 48
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Presented at the Fourth Army Conference on Applied Mathematics and Computing, Cornell University, Ithaca, New York, 27-30 May 1986.
Published in the Proceedings of the Conference.

(CONT'D ON REVERSE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Adaptive Methods
Moving Mesh Methods
Local Refinement
Hyperbolic Systems
Partial Differential Equations

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We discuss mesh moving, static mesh regeneration, and local mesh refinement algorithms that can be used with a finite difference or finite element scheme to solve initial-boundary value problems for vector systems of time-dependent partial differential equations in two-space dimensions and time. A coarse base mesh of quadrilateral cells is moved by an algebraic mesh movement function so that it may follow and isolate spatially distinct phenomena. The local mesh refinement method recursively divides the time step and spatial cells of the

(CONT'D ON REVERSE)

7. AUTHORS (CONT'D)

David C. Arney
Department of Mathematics
United States Military Academy
West Point, NY 10996-1786

Joseph E. Flaherty
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

and

US Army ARDEC
Close Combat Armaments Center
Benet Laboratories
Watervliet, NY 12189-4050

18. SUPPLEMENTARY NOTES (CONT'D)

This research was partially supported by the U.S. Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant Number AFOSR 85-0156 and the U.S. Army Research Office under Contract Number DAAL 03-86-K-0112. This work was used to partially fulfill the Ph.D. requirements of the first author at Rensselaer Polytechnic Institute.

20. ABSTRACT (CONT'D)

moving base mesh in regions where error indicators are high until a prescribed tolerance is satisfied. The static mesh regeneration procedure is used to create a new base mesh when the existing ones become too distorted.

In order to test our adaptive algorithms, ~~we~~ implemented them in a system code with an initial mesh generator, a MacCormack finite difference scheme for hyperbolic systems, and an error indicator based upon estimates of the local discretization error obtained by Richardson extrapolation. Results are presented for several computational examples.

TABLE OF CONTENTS

Page

I. INTRODUCTION.

Many initial-boundary value problems for time-dependent partial differential equations involve fine-scale structures that develop, propagate, decay, and/or disappear as the solution evolves. Some examples are shock waves in compressible flows, boundary and shear layers in viscous flows, and reaction zones in combustion processes. The numerical solution of these problems is usually difficult because the nature, location, and duration of the structures are often not known in advance. Thus, conventional numerical approaches that calculate solutions on a prescribed (typically uniform) mesh often fail to adequately resolve the fine-scale phenomena, have excessive computational costs, or produce incorrect results. Adaptive procedures that evolve with the solution offer a robust, reliable, and efficient alternative. Such techniques have been the subject of a great deal of recent attention (cf. Babuska et al. [7, 9])* and are generally capable of introducing finer meshes in regions where greater resolution is needed [1, 2, 3, 6, 8, 10, 15, 16], moving meshes in order to follow isolated dynamic phenomena [1, 2, 5, 21, 23, 24, 25, 30], or changing the order of methods in specific regions of the problem domain [18, 22]. The utility of such adaptive techniques is greatly enhanced when they are capable of providing an estimate of the accuracy of the computed solution. Local error estimates are often used as refinement indicators and to produce solutions that satisfy either local or global accuracy specifications [1, 2, 3, 6, 8, 10, 15, 16]. Successful error estimates have been obtained

* References are listed at the end of this report.

using h-refinement [6, 15, 16], where the difference between solutions on different meshes is used to estimate the error, and p-refinement [1, 2, 3, 8, 16, 22] where the differences between methods of different orders are used to estimate the error.

We discuss an adaptive procedure that combines mesh movement and local refinement for m-dimensional vector systems of partial differential equations having the form

$$\mathbf{u}_t + \mathbf{f}(x,y,t,\mathbf{u},\mathbf{u}_x,\mathbf{u}_y) = [\mathbf{D}^1(x,y,t,\mathbf{u})\mathbf{u}_x]_x + [\mathbf{D}^2(x,y,t,\mathbf{u})\mathbf{u}_y]_y, \quad \text{for } t > 0, \quad (x,y) \in \Omega, \tag{1a}$$

with initial conditions

$$\mathbf{u}(x,y,0) = \mathbf{u}^0(x,y), \quad \text{for } (x,y) \in \Omega \cup \partial\Omega, \tag{1b}$$

and appropriate well-posed boundary conditions on the boundary $\partial\Omega$ of a rectangular region Ω.

We suppose that a numerical method is available for calculating approximate solutions and error indicators of Eq. (1) at each node of a moving mesh of quadrilateral cells. Any appropriate numerical method is applicable and the error indicator can either be an estimate of the local discretization error or another function (e.g., an estimate of the solution gradient or curvature) that is large where additional resolution is needed and small where less resolution is desired. Our adaptive algorithm consists of three main parts: (a) movement of a coarse base mesh, (b) local refinement of the base mesh in regions where resolution is inadequate, and (c) creation and regeneration of the base mesh when it becomes overly distorted. Our experience (cf. Section III) and that of others [25] indicates that mesh motion can substantially reduce errors for a very modest computational cost. Mesh motion alone, however, cannot produce a solution that will satisfy a prescribed error tolerance in all situations. For this reason, we have combined mesh motion with local mesh refinement, and recursively solve local problems in regions where error tolerances are not satisfied. The local solution scheme successively reduces the domain size and, thus, further reduces the cost of the computation. Some problems, e.g., those with severe material deformations, can result in tangling and distortion of the moving base mesh. Therefore, we have created a procedure that automatically generates a new base mesh whenever the old one is unsuitable.

The adaptive procedures described in this report combine our earlier work on mesh moving techniques [5] and local refinement procedures [6]. The inclusion of a static mesh regeneration scheme adds greater reliability and efficiency to these methods. The three components of our adaptive algorithm are described in Section II; however, frequent references are made to our previous investigations [5, 6]. A computer code based on the adaptive algorithm of Section II has been combined with a MacCormack finite difference scheme and an error indicator based on Richardson extrapolation. It has been used to solve a sequence of hyperbolic problems (i.e., problems having the form of Eq. (1) with $\mathbf{D}^1 = \mathbf{D}^2 := 0$) and our findings on three examples, where we have attempted to appraise the relative costs and benefits of the mesh moving and local refinement portions of our adaptive algorithm, are reported in Section III. We have also compared solutions obtained by adaptive techniques to those obtained using stationary uniform meshes. In all three examples, solutions obtained by adaptive techniques cost less than solutions obtained on stationary uniform meshes having approximately the same accuracy. The mesh moving technique added approximately ten percent to the computational time of the adaptive algorithm and greatly improved the results. Most of the computational time was devoted to calculating the solution and error indicators, and not to the overhead induced by the refinement procedure. Although we are greatly encouraged by our results, our adaptive procedures are far from complete. Some possible improvements and future considerations are discussed in Section IV.

II. ALGORITHM DESCRIPTION.

A top-level description of our adaptive procedure is presented in Figure 1 in a pseudo-PASCAL language. This procedure is called *adaptive_PDE_solver* and it integrates a system of partial differential equations from time *tinit* to *tfinal* and attempts to keep the local error indicators below a tolerance of *tol*. The base level time step Δt is initially specified, but may be changed, as needed, during the integration.

The rectangular domain Ω is initially discretized into a coarse moving spatial grid of $M \times N$ quadrilateral cells. An initial base mesh is generated from this mesh by increasing the values of M and N, as necessary, and moving the mesh so that it is concentrated in

```
procedure adaptive_PDE_solver($tinit$, $\Delta t$, $tfinal$, $tol$: real; $M$, $N$: integer);

  begin
    Generate an initial base mesh;
    $t := tinit$;

    while $t < tfinal$ do
      begin
        Move the base mesh for the time step $t$ to $t + \Delta t$;
        local_refine(0, $t$, $\Delta t$, $tol$);
        $t := t + \Delta t$;
        Select an appropriate $\Delta t$;
        if base mesh is too distorted then regenerate a base mesh
      end
  end { adaptive_PDE_solver };
```

Figure 1. Pseudo-PASCAL description of an adaptive procedure to solve the partial differential system in Eq. (1) from $t = tinit$ to $tfinal$ to within a tolerance of tol.

regions where error indicators are large (cf. Section II.3). The base mesh is moved for each base time step Δt (cf. Arney and Flaherty [5] and Section II.1) and the partial differential system in Eq. (1) is solved on this mesh for a base time step. This is followed by a recursive local mesh refinement in regions where error indicators are larger than tol. The local refinement procedure *local_refine* was described in Arney and Flaherty [6] and its major features are summarized in Section II.2. The integration for each base-mesh time step is concluded by the selection of a new value of Δt for the subsequent time step and the generation of new base mesh (cf. Section II.3), if necessary.

The mesh moving, local refinement, and mesh regeneration algorithms are uncoupled from each other as well as from the procedures used to solve the partial differential system and calculate local error indicators. This reduces computational costs and provides a great deal of flexibility. Thus, individual modules can easily be replaced, omitted, or combined with other software.

II.1. Mesh Moving Algorithm. Mesh moving strategies should produce a smooth mesh where the sizes of neighboring computational cells vary slowly and cell angles differ only by modest amounts from right angles. It is, of course, essential for the nodes of the mesh to remain within Ω and for cells not to overlap. Meshes that violate these conditions can produce large discretization errors that overwhelm the positive effects of mesh moving. Our mesh moving procedure is based on an intuitive approach rather than more analytic error equidistribution (cf., e.g., Coyle et al. [19] or Dwyer [23]) and variational approaches (cf. Brackbill and Saltzman [17]). The essential idea is to move the mesh so that it may follow isolated nonuniformities, such as wave fronts, shock layers, and reaction zones. This generally reduces dispersive errors and allows the use of larger time steps while maintaining accuracy and stability.

At each base time, we scan the $M \times N$ base mesh of quadrilateral cells and locate "significant error nodes" as those having error indicators greater than twice the mean nodal error indicator and also greater than ten percent of tol. This empirical strategy avoids having the mesh respond to fluctuations when error indicators are too small, but is sensitive enough to avoid missing dynamic phenomena associated with large error indicators. If there are no significant error nodes, computation proceeds on a stationary mesh. The nearest neighbor clustering algorithm of Berger and Oliger [15] is then used to gather the significant error nodes into clusters. In this iterative algorithm, a cluster is first defined to consist of one arbitrary significant error node. Other significant error nodes are added to the cluster if they are within a specified minimum intercluster distance from the nearest node in the cluster. We take the minimum intercluster distance to be the length of a cell diagonal. New clusters are established for nodes that do not belong to any existing cluster. Clusters are united when a node is determined to belong to more than one of them. Upon completion of the algorithm, (a) nodes in different clusters will be separated by at least the minimum intercluster distance, and (b) no node in a cluster with more than one node will be further than the minimum intercluster distance from its nearest neighbor in the cluster.

Following Berger and Oliger [15], we generate near minimum area rectangles that contain each cluster. The principal axes of each rectangle are the major and minor axes of an enclosed ellipse having the same first and second moments as the nodes in the cluster. Thus, if (x_i, y_i) are the coordinates of a node and (x_m, y_m) are the mean coordinates of all nodes in the cluster, then the axes of the rectangle are in the directions of the eigenvectors of the symmetric (2×2) matrix

$$\begin{bmatrix} \sum (x_i^2 - x_m^2) & \sum (x_i y_i - x_m y_m) \\ \sum (x_i y_i - x_m y_m) & \sum (y_i^2 - y_m^2) \end{bmatrix}, \tag{2}$$

where the summations range over all nodes in the cluster.

For many problems, there may be too small a percentage of significant error nodes within a cluster. In order to reduce this inefficiency and provide some alignment with, e.g., curved wave fronts, the rectangles are checked for efficiency by determining the percentage of significant error nodes in each rectangle. If a fifty-percent efficiency is not achieved, the rectangle is iteratively bisected in the direction of its major axis until all clusters have at least a fifty-percent efficiency.

We determine node movement from the velocity of propagation, the orientation, and the size of error clusters. We assume that nodes in the same cluster have related solution characteristics, so that we can determine individual node movement from the propagation of the center of the error cluster. Each cluster moves according to the differential equation

$$\ddot{\mathbf{r}}_m + \lambda \dot{\mathbf{r}}_m = 0, \tag{3}$$

where $\mathbf{r}_m(t) = [x_m(t), y_m(t)]^T$ is the position of the center of an error cluster and $(\dot{\ }) := d(\ )/dt$. The choice of the parameter λ can be critical in certain situations. If λ is selected too large, the system in Eq. (3) will be stiff and computationally expensive. On the other hand, if λ is too small, the mesh can oscillate from time step-to-time step. Coyle et al. [19] and Adjerid and Flaherty [2] suggested some adaptive procedures for choosing λ; however, we found no appreciable differences in results or computation times when λ varied significantly. The examples of Section III were calculated with $\lambda = 1$.

We solve Eq. (3) for each base time step and each cluster using an explicit numerical method. The center of an error cluster is moved a distance $\Delta \mathbf{r}_m = \mathbf{r}_m(t+\Delta t) - \mathbf{r}_m(t)$ at the base time t. Let Δr_{m_1} and Δr_{m_2} denote the projections of $\Delta \mathbf{r}_m$ onto the major and minor axes of the rectangular cluster. We use the one-dimensional piecewise linear function

$$d_{i,inside} = \begin{cases} \Delta r_{m_i}(3/2 + x_i/w_i), & \text{if } -3w_i/2 \le x_i \le -w_i/2 \\ \Delta r_{m_i}, & \text{if } -w_i/2 < x_i < w_i/2 \\ \Delta r_{m_i}(3/2 - x_i/w_i), & \text{if } w_i/2 \le x_i \le 3w_i/2 \\ 0, & \textit{otherwise} \end{cases} , \quad i = 1, 2, \tag{4}$$

to move the nodes of the mesh along the two principal axial directions of the error clusters. The cluster referred to in Eq. (4) has dimensions $w_1 \times w_2$, and (x_1, x_2) are local Cartesian coordinates of a node in the principal directions of the cluster relative to its center. For $i = 1$, nodes in the range of the cluster $(-3w_1/2 \le x_1 \le 3w_1/2, -w_2/2 \le x_2 \le w_2/2)$ are moved a distance $d_{1,inside}$. This situation is shown in Figure 2.

In order to maintain smooth mesh motion throughout the domain, nodes outside the range of a cluster move in a distance

$$d_{i,outside} = d_{i,inside}[1 - (2z/D)], \quad i = 1, 2, \tag{5}$$

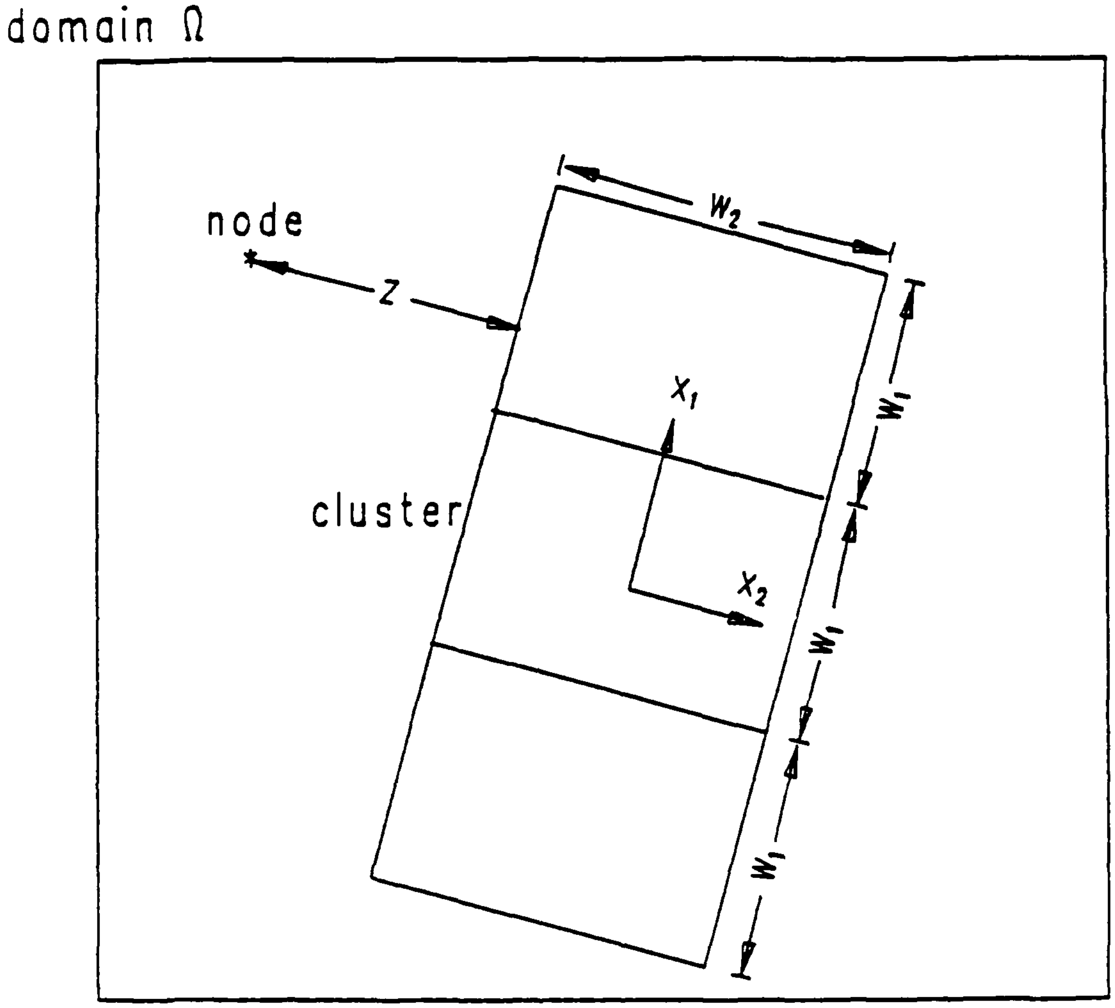

Figure 2. A rectangular $w_1 \times w_2$ error cluster. Nodes within the range of the cluster, $3w_1 \times w_2$, are moved a distance $d_{1,inside}$ in the x_1 principal direction according to Eq. (4). Nodes outside the range of the cluster are moved a distance $d_{1,outside}$ in the x_1 direction according to Eq. (5). The distance z is the shortest distance to the range of the cluster.

where z is the shortest distance to the range of the cluster (cf. Figure 2) and D is the diagonal of Ω. For each cluster, the mesh is moved in the direction of the major axis ($i = 1$) using Eqs. (4) and (5). This is followed by a similar procedure in the direction of the minor axis ($i = 2$). The distances $d_{i,inside}$ and $d_{i,outside}$ are reduced near $\partial\Omega$ in order to prevent nodes from leaving Ω. In particular, we recalculate $d_{i,j}$ as $d_{i,j}[min(1, b/c)]$, $i = 1, 2$, $j = inside, outside$, where b is the distance of the node to the boundary and c is twice the length of a cell diagonal on a uniform mesh having the same number of cells as the moving mesh. Nodes on domain boundaries, except corner nodes, which are not moved, are restrained to move along the boundary. Finally, the mesh moving algorithm is not restricted to the functions given by Eqs. (4) and (5), and several other choices are possible.

II.2. Local Refinement Algorithm. As shown in Figure 1, the local refinement procedure is invoked after the base mesh has been moved for a base time step. Our refinement strategy consists of first calculating a preliminary solution on the base mesh for a base time step. An error indicator is used to locate regions where greater resolution is needed. Finer grids are adaptively created in these high-error regions by locally bisecting the time step and the sides of the quadrilateral cells of the base grid and the solution and error indicators are computed on the finer grids. The refinement scheme is recursive; thus, fine subgrids may be refined by adaptively creating even finer subgrids. This relationship leads naturally to a tree-data structure. Information regarding the geometry, solution, and error indicators of the base grid is stored as the root node or level 0 of the tree. Subgrids of the base grid are offsprings of the root node and are stored as level 1 of the tree. The structure continues, with a grid at level l having a parent coarser grid at level $l - 1$ and any finer offspring grids at level $l + 1$. Grids at level l of the tree are given an arbitrary ordering and we denote them as $G_{l,j}$, $j = 1, 2, ..., N_l$, where N_l is the number of grids at level l. Our refinement procedures permit grids at the same level of a two-dimensional problem to intersect and overlap; however offspring grids must be properly nested within the boundaries of their parent grid. A one-dimensional grid with its appropriate tree structure for a base time step is shown in Figure 3.

A top-level pseudo-PASCAL description of a recursive local refinement algorithm that solves systems of the form in Eq. (1) on the tree of grids described above is presented in Figure 4. The procedure local_refine integrates partial differential equations on the grids $G_{l,j}$, $j = 1, 2, ..., N_l$, at level l of the tree from time *tinit* to *tinit* + Δt and attempts to satisfy a prescribed local error tolerance *tol*. For each grid at level l, a solution and error indicators are calculated at time *tinit* + Δt. Additional finer grids are introduced in regions where the error indicators exceed the prescribed tolerance *tol* and the differential system is solved again on the finer grids using two time steps of duration $\Delta t/2$ and a tolerance of *tol*/2. Observe that the solution, error indicators, and refined subgrids are calculated for all grids at level l before calculating any solutions at level $l + 1$. Implicit in local_refine are the assumptions that a solution can be computed on any grid and that refinement terminates. If either of these assumptions are violated, the procedure terminates in failure.

Our technique for introducing finer subgrids consists of four steps: (a) an initial scan

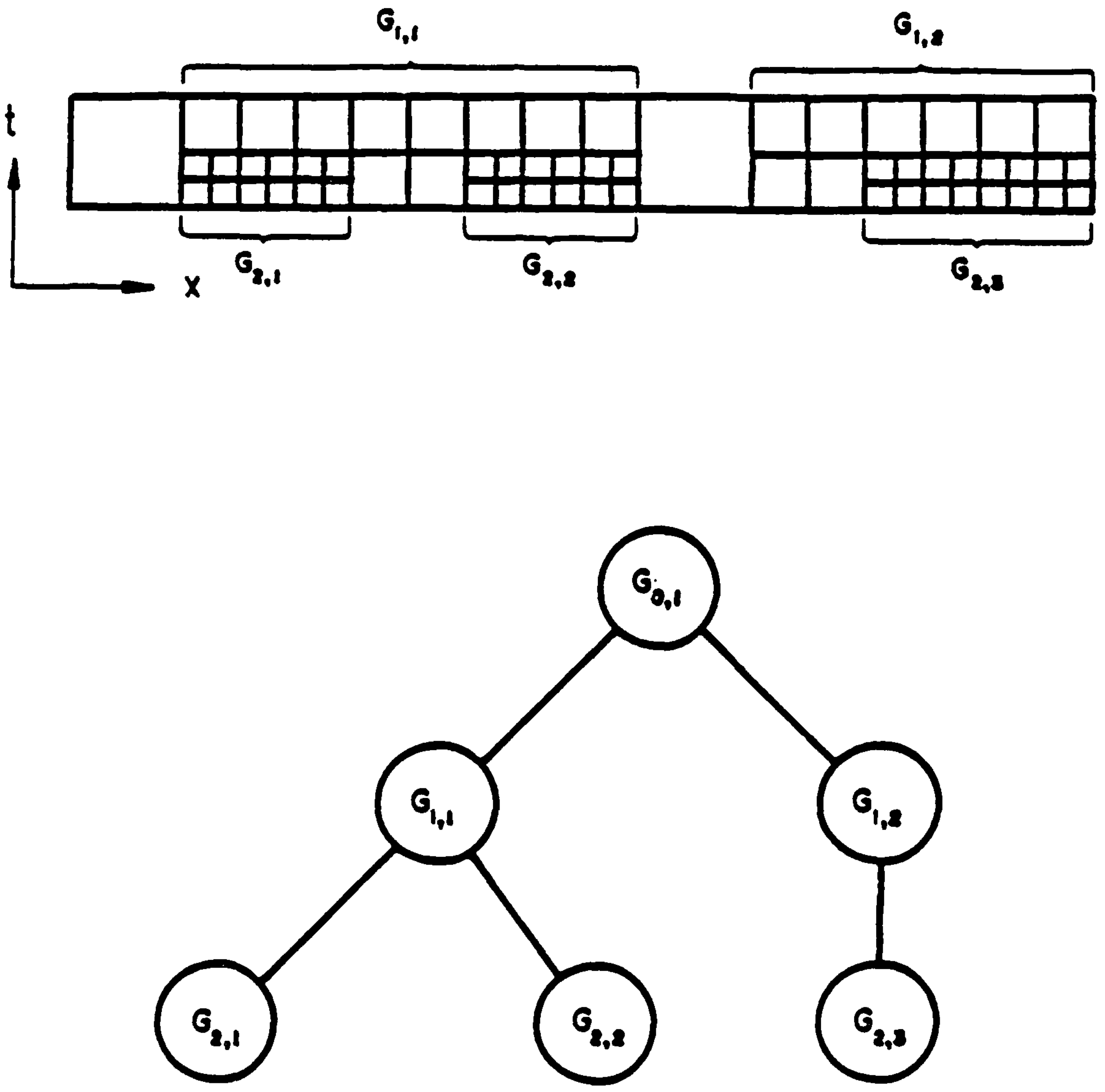

Figure 3. Coarse and refined grids (top) and their tree representation (bottom) for a one-dimensional example.

of each level l grid to locate "untolerable-error" nodes as those where the error indicator exceeds the prescribed tolerance tol, (b) clustering any untolerable nodes into rectangular regions, (c) buffering the clustered regions in order to reduce problems associated with

```
procedure local_refine( $l$: integer; $tinit$, $\Delta t$, $tol$: real );

  begin
    for $j$ := 1 to $N[l]$ do
      begin
        Integrate the partial differential system from $tinit$ to $tinit + \Delta t$
          on grid $G[l,j]$;
        Calculate error indicators at $tinit + \Delta t$ at all nodes of grid
          $G[l,j]$;
        if any error indicators > $tol$ then introduce level $l + 1$ subgrids
          of $G[l,j]$
      end { for };

    if any error indicators > $tol$ then
      begin
        local_refine($l + 1$, $tinit$, $\Delta t/2$, $tol/2$);
        local_refine($l + 1$, $tinit + \Delta t/2$, $\Delta t/2$, $tol/2$)
      end
  end { local_refine };
```

Figure 4. Pseudo-PASCAL description of a recursive local refinement procedure to find a solution of the partial differential system in Eq. (1) on all grids at level l of the tree.

prescribing initial and boundary conditions at coarse/fine grid interfaces, and (d) cellularly refining the level l meshes and time step within the buffered clusters. Of course, if there are no untolerable-error nodes, the solution is acceptable and further refinement is unnecessary.

The same clustering algorithm of Berger and Oliger [15] that was used to move the base mesh is also used to group untolerable-error nodes for refinement. Each rectangular error cluster is enlarged by increasing its major and minor axes by twice the size of the average cell edge within the cluster. The region between the enlarged and original error clusters provides a buffer so that artificial internal boundary conditions (that are discussed below) will be prescribed at low-error nodes as far as possible and fine-grid errors will not propagate through the buffer in a time step.

Refined subgrids are created by bisecting the time step and edges of each cell of the parent mesh that intersects the buffered rectangular error clusters. Coarse mesh motion is maintained on the refined grids so that after two time steps of size $\Delta t/2$, cells of the refined grids will be properly nested within those of their parent grid. Additional details of the refinement algorithm and data structures are presented in Arney and Flaherty [6].

Artificial initial and boundary data must be determined from solutions on other grids in order to calculate the solution and error indicators on refined subgrids. Furthermore, solutions on finer grids are used to replace those on coarser grids at common nodal locations.

Initial data for a subgrid are calculated directly from the initial function $\mathbf{u}^0(x,y)$ at $t = 0$. For $t > 0$, initial data are obtained by interpolation using the solution at the same time on the finest available mesh. In order to provide data for this interpolation, we save all solution values on previous subgrids until they are no longer needed due to advancement in time of an acceptable solution. Bilinear functions using the solution values at the four vertices of the finest existing cell are used to obtain the solution at the nodes of cells of the refined mesh. Further analysis is needed regarding the effects on accuracy and stability and the proper order of this interpolation. Bieterman, Flaherty, and Moore [16] give an example where the fine-scale structure of a solution was lost by interpolation from too coarse a mesh.

In a similar manner, boundary data for refined meshes are calculated directly from the prescribed boundary conditions on portions of subgrids that intersect $\partial\Omega$. Dirichlet boundary data are prescribed on the edges of subgrids that are in the interior of Ω by interpolating the solution from coarser meshes. Bilinear functions using the solution values at the four vertices of the adjacent face of the finest existing space-time cell are used to obtain solution values for the nodes of refined cells.

Acceptable fine-mesh solutions are used to replace solutions at the nodes of coarser grids that lie within the untolerable-error portions of clusters. Solutions at low-error nodes in the buffer zones of clusters are not replaced in order to avoid possible contamination of accurate solutions. When fine grids overlap each other in an untolerable-error region, the average value of the solutions at common fine-grid nodes is used to replace the appropriate coarse grid solution. Boundary effects do not propagate through a sufficiently large buffer and, thus, have no effect on the solution within the untolerable-error region of a cluster when an explicit numerical scheme is used for the integration. Greater care is needed when implicit integration methods are used, since artificial boundary conditions can affect the accuracy, convergence, and stability of the solution at all nodes in the cluster regardless of the size of the buffer.

Stability and conservation of, e.g., fluxes at interfaces between coarse and fine meshes must be investigated further, particularly in two dimensions. For one-dimensional problems, Berger and Oliger [15] showed that linear interpolation of solutions from a coarse to a fine mesh produced no instabilities in the the Lax-Wendroff scheme. Berger [14] also discussed conservation at mesh interfaces and proposed explicit enforcement of conserved quantities at coarse/fine mesh boundaries. Rai [29] presented some finite difference schemes that maintained conservation at grid interfaces for two-dimensional compressible flow problems.

II.3. Initial Mesh Construction and Regeneration. The efficiency of our adaptive mesh moving and refinement strategies is dependent on our ability to generate a suitable initial mesh and to regenerate a new base mesh should it become severely distorted at later times. The proper base mesh can reduce the need for refinement and, thus, increase efficiency.

The two essential elements of a mesh generation or regeneration procedure are the determination of the number of nodes and their optimal location. A base mesh having too few nodes will result in excessive refinement while one having too many nodes will reduce efficiency. Many mesh generation procedures have been developed (cf., e.g., Thompson [31] or Brackbill and Saltzman [17]); however, the best one to use in conjunction with an adaptive procedure is still far from being established. Our current approach to mesh generation is to use the error indicators computed by a trial solution to determine an initial mesh that approximately equidistributes the error indicators.

To begin, we create a uniform $M \times N$ rectangular mesh using prescribed values of M and N that reflect the coarsest mesh that should be used to calculate a solution. We solve the system in Eq. (1) for a base time step Δt on the uniform stationary base mesh and compute the solution and error indicators. Local mesh refinement is performed as described in Section II.2 until the prescribed tolerance is attained. We use this solution to determine the number of nodes K in a new base mesh as

$$K = MN + \sum_{l=1}^{n} (3/4)^l K_l, \tag{6a}$$

where K_l is the number of nodes introduced at level l and n is the total number of levels in the tree. Having computed K, we calculate the dimensions of a new $\overline{M} \times \overline{N}$ mesh as

$$\overline{M} = \sqrt{KM/N}, \quad \overline{N} = \sqrt{KN/M}. \tag{6b}$$

The bars have been omitted on M and N in the algorithms displayed in Figures 1 and 4 and in all further discussions.

Node placement for the new base mesh is accomplished by locating all nodes of the original base mesh having error indicators that are greater than twice the mean error indicator. These nodes are then grouped into rectangular clusters using the clustering algorithm of Section II.1. A uniform base mesh is generated when there are no nodes having error indicators that are greater than twice the mean error indicator.

Nodes are moved towards the center of the nearest error cluster unless they are within a two-cell diagonal range of two or more error clusters. In the former case, a node is moved four-tenths of its distance to the center of the nearest cluster unless this distance is greater than 12.5 times the average cell diagonal, in which case it is moved five times the average cell diagonal. Nodes that are within a two-cell diagonal range of two or more clusters are moved by four-tenths of a weighted average of the distances to centers of the involved clusters. Nodes on $\partial\Omega$ remain on $\partial\Omega$. Nodes near the boundary move a reduced distance in order to prevent the formation of large elements. When an error cluster intersects opposite boundaries of Ω, nodes are not moved in the direction of the major axis of the cluster. This construction generates a base mesh that depends on the solution of the partial differential system as well as its initial condition.

The base mesh can become severely distorted for some problems (cf. Arney and Flaherty [5]) and we would like a capability for generating a new base mesh whenever this happens. Since the new mesh is created at a specific time, rather than by mesh motion, we refer to this process as static mesh regeneration. Our static mesh regeneration procedure consists of three steps: (a) determining that there is a need for a new base mesh, (b) creating the new base mesh, and (c) interpolating the solution from the old to the new base mesh.

A mesh is regenerated when any interior angle of a cell is less than 50 or greater than 130 degrees, the aspect ratio of any cell is greater than 15, or the mesh ratio of adjacent cells exceeds 5 or is less than 1/5. In the present context, the aspect ratio is defined as the average length divided by the average width of a cell and the mesh ratios are defined as the ratio of the lengths and widths of adjacent cell sides.

A new base mesh, having the same number of nodes as the old one, is generated using the procedure described above for creating an initial base mesh. The error clusters for the existing mesh are used to generate the new base mesh, so that new clusters do not have to be computed. This process appears to reduce angle deviations from ninety degrees, control aspect ratios, and mollify adjacent mesh ratios.

Once a new base mesh has been constructed, the solution on the old one is interpolated to the new one by using bilinear interpolation with respect to the cells of the old base mesh. The order and nature of the interpolation needs further investigation and we are studying methods that conserve, e.g., fluxes (cf. Berger [14] or Rai [29]).

III. COMPUTATIONAL EXAMPLES.

In order to demonstrate the capabilities of the adaptive procedure described in Section II, we applied it to three hyperbolic systems. We used a two-step MacCormack finite difference method (cf. Arney and Flaherty [5], Hindman [26], or MacCormack [27]) to integrate the partial differential equations and Richardson's extrapolation (cf. Arney [4] or Berger and Oliger [15]) to indicate local errors. Base mesh geometry was prescribed as indicated in each example. If the base mesh time step failed to satisfy the Courant, Friedrichs, Lewy theorem, it was automatically reduced to the maximum allowed by the Courant condition (cf. Arney [4] and Arney and Flaherty [6]). This procedure should also satisfy the Courant condition on all subgrids when the characteristic speeds vary slowly.

Numerical results obtained on uniform stationary grids are compared with those obtained by adaptive strategies that use (a) mesh moving only, (b) local refinement only, and (c) the combination of mesh moving and refinement discussed in Section II. The examples are designed to determine the relative cost, accuracy, and efficiency of our adaptive algorithm and each of its components. Accuracy is appraised by computing the difference e between the exact and numerical solutions of a problem in either the maximum or L_1 norms, i.e., by computing either

$$\|\mathbf{e}(\cdot,\cdot,t)\|_\infty := \max_{1\le i\le K} \max_{1\le j\le m} |e_j(x_i, y_i, t)|, \tag{7a}$$

or

$$\|\mathbf{e}(\cdot,\cdot,t)\|_1 = \iint_\Omega P \sum_{j=1}^{m} |e_j| \, dxdy, \tag{7b}$$

respectively. Here, K is the number of nodes in the mesh at time t and P is a piecewise constant interpolation operator with respect to the cells of the base mesh that, on each cell, has the average value of the errors at the vertices of the cell. We use either the total CPU time or the maximum number of nodes used in a base time step as measures of the computational complexity of a procedure. All calculations were performed in double precision arithmetic on an IBM 3081/D computer at the Rensselaer Polytechnic Institute.

Solutions are displayed by drawing either level lines or wire-frame perspective renditions. Meshes are displayed by showing the complete two-dimensional spatial discretization at specified times with finer subgrids overlaying coarser ones. This portrayal does not show the reduced time steps that are used for the subgrid calculations. The broken-line rectangles in the figures indicate the error cluster(s) that are used to move the base mesh.

Example 1. Consider the linear initial-boundary value problem that was proposed as a test problem by McRae et al. [28]:

$$u_t - yu_x + xu_y = 0, \quad t > 0, \quad (x,y) \in \Omega, \tag{8a}$$

$$u(x,y,0) = \begin{cases} 0, & \text{if } (x-1/2)^2 + 1.5y^2 \geq 1/16 \\ 1 - 16((x-1/2)^2 + 1.5y^2), & \textit{otherwise}, \end{cases} \quad (x,y) \in \Omega \cup \partial\Omega, \tag{8b}$$

and

$$u(x,y,t) = 0, \quad t > 0, \quad (x,y) \in \partial\Omega, \tag{8c}$$

where $\Omega := \{ (x,y) \mid -1.2 < x, y < 1.2 \}$.

The exact solution of Eq. (8) is an elliptical cone that rotates about the origin in the counterclockwise direction with period 2π. It can be written in the form

$$u(x,y,t) = \begin{cases} 0, & \text{if } C < 0 \\ C, & \text{if } C \geq 0, \end{cases} \tag{9a}$$

where

$$C = 1 - 16[(x\cos t + y\sin t - 1/2)^2 + 1.5(y\cos t - x\sin t)^2]. \tag{9b}$$

Five adaptive and uniform mesh solutions of Eq. (8) were calculated for $0 < t \leq 3.2$ and our findings are summarized in Table 1. Solutions 3 and 4, with refinement, were calculated using an error tolerance of 0.0002 and a maximum of two levels of refinement. The tolerance and maximum level of refinement were selected so that the high-error region under the cone would maintain approximately the same mesh spacing as the uniform mesh used to obtain Solution 5. The grids that were used to obtain Solution 4 are shown in Figure 5 at t = 0.56, 1.68, 2.24, and 3.2. A new base mesh was introduced at t = 2.82. The meshes that were used to obtain Solutions 2, 3, and 4 at t = 3.2 are shown in Figure 6. Finally, surface and contour plots of Solutions 1, 2, and 3 and of Solutions 4 and 5 at t = 3.2 are shown in Figures 7 and 8, respectively.

Solution 1 bears no resemblance to the exact solution and demonstrates the devastating effects of large dissipative anu dispersive errors. Solution 2, with mesh moving only, provides a dramatic improvement in the results for approximately one-half the cost of using both mesh motion and refinement. Solution 5 took more than three-times longer to calculate than Solution 4 for approximately the same accuracy; thus, demonstrating the efficiency of the refinement process. The subgrids for the refined Solutions 3 and 4 are concentrated in the region of the cone and are aligned with its principal axes as it rotates.

Ref. No.	Strategy	Base Mesh	Base Time Step	$\|e\|_1$	$\|e\|_\infty$	CPU Time (sec.)
1	Stationary uniform mesh	14×14	0.056	0.2560	0.78	46
2	Moving mesh	32×32	0.026	0.0301	0.20	458
3	Stationary mesh with refinement	14×14	0.056	0.0832	0.48	852
4	Moving mesh with refinement	14×14	0.056	0.0249	0.18	904
5	Stationary uniform mesh	56×56	0.014	0.0759	0.48	2647

Table 1. Errors at t = 3.2 and computational costs for five solutions of Example 1.

Dissipative and dispersive errors cause a "wake" of spurious oscillatory information to follow the moving cone (cf. Figures 7 and 8). Some mesh refinement is performed in the wake region and this greatly reduces the magnitude of the oscillations.

Example 2. Consider the uncoupled linear initial-boundary value problem

$$u_{1_t} + u_{1_x} = 0, \quad u_{2_t} - u_{2_x} = 0, \quad t > 0, \quad (x,y) \in \Omega, \tag{10a}$$

$$u_1(x,y,0) = \begin{cases} 1 - 16((x-1/2)^2 + 1.5y^2), & \text{if } (x-1/2)^2 + 1.5y^2 \le 1/16 \\ 0, & \textit{otherwise}, \end{cases}$$
$$(x,y) \in \Omega \cup \partial\Omega, \tag{10b}$$

$$u_2(x,y,0) = \begin{cases} 1 - 16((x+1/2)^2 + 1.5y^2), & \text{if } (x+1/2)^2 + 1.5y^2 \le 1/16 \\ 0, & \textit{otherwise}, \end{cases}$$
$$(x,y) \in \Omega \cup \partial\Omega, \tag{10c}$$

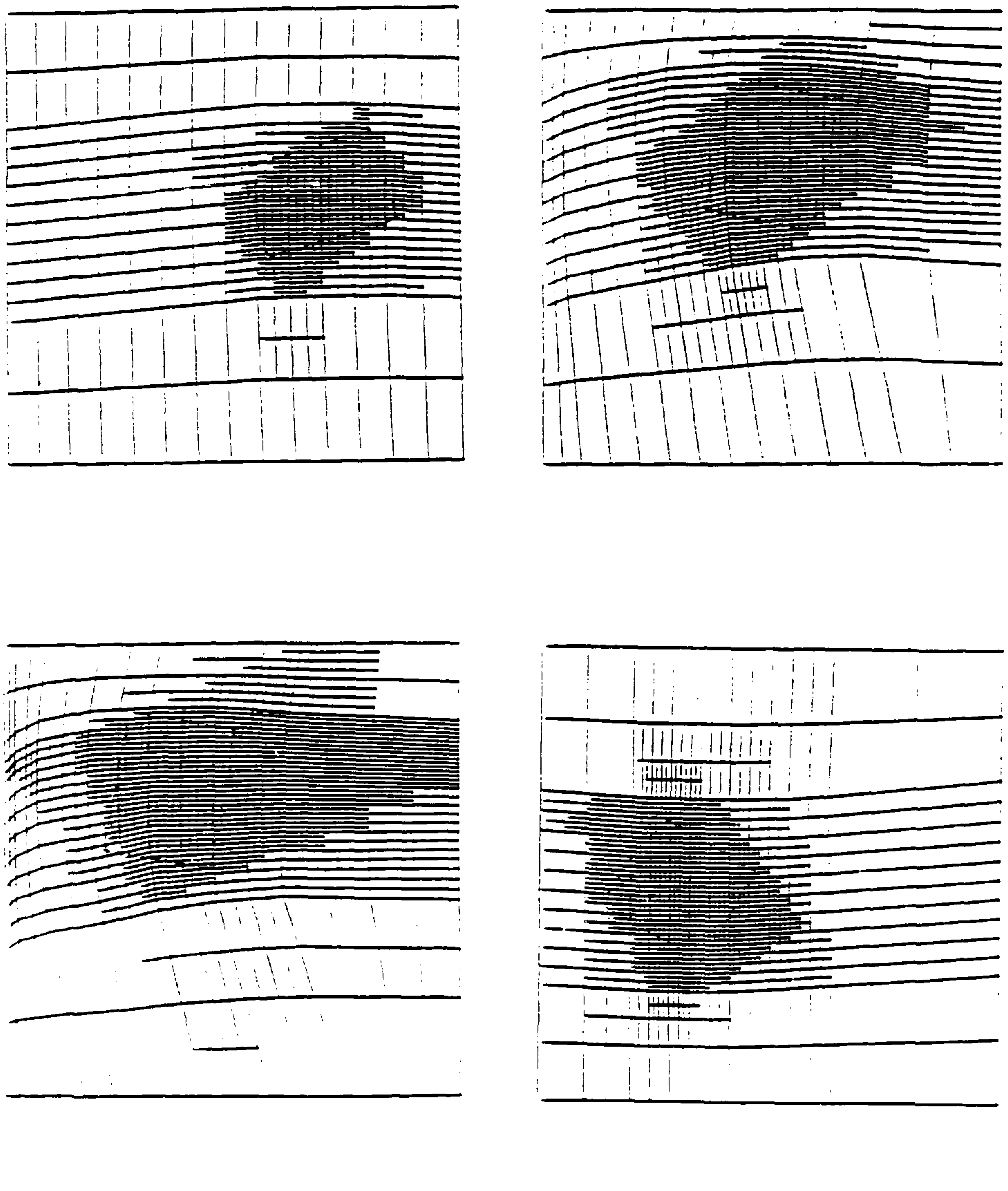

Figure 5. Grids created for Solution 4 of Example 1 at t = 0.056 (upper left), 1.68 (upper right), 2.24 (lower left), and 3.2 (lower right).

$$u_1(x,y,t) = u_2(x,y,t) = 0, \quad t > 0, \quad (x,y) \in \partial\Omega, \tag{10d}$$

and $\Omega := \{ (x,y) \mid -1 \le x \le 1, -0.6 \le y \le 0.6 \}$.

The solution of this problem consists of two moving cones that collide and pass through each other. We selected it in order to determine how the various adaptive strategies could cope with interacting phenomena.

One uniform mesh and three adaptive solutions of Eq. (10) were calculated for $0 < t \le 1.2$ and our findings are summarized in Table 2. The solutions involving refinement were computed with a tolerance of 0.0038. All solutions were designed to have approximately the same accuracy. The grids that were used to obtain Solution 4 are shown in Figure 9 at $t = 0, 0.23, 0.46, 0.92$, and 1.2.

The results of Table 2 demonstrate the efficiency of the mesh moving strategy on this example. Solution 2 with mesh moving was slightly more accurate than Solution 1 obtained on a uniform mesh, and it required less than one-half of the computation time. Solution 3 with refinement on a stationary mesh shows only a modest improvement over Solution 1; however, the combination of mesh moving and refinement computed in Solution 4 again shows a significant gain in efficiency. We suspect that the high accuracy achieved by mesh moving on this example is due to the reduction in dispersive errors that results when the mesh follows the cones with approximately the correct velocity.

Example 3. Consider the Euler equations for a perfect inviscid compressible fluid

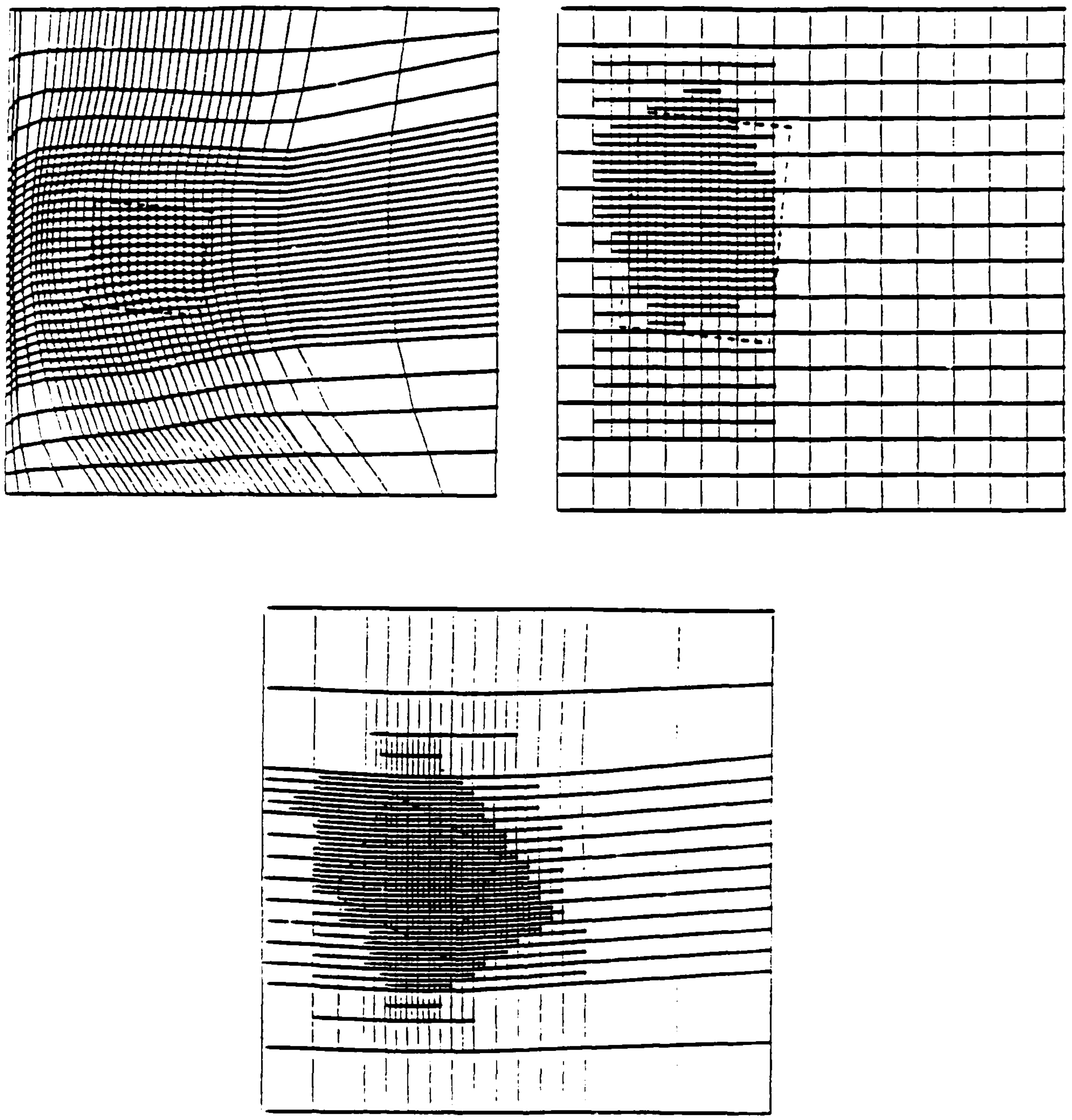
Figure 6. Grids created for Solutions 2, 3, and 4 of Example 1 at $t = 3.2$.

$$\mathbf{u}_t + \mathbf{f}_x(\mathbf{u}) + \mathbf{g}_y(\mathbf{u}) = 0, \tag{11a}$$

where

$$\mathbf{u} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho u^2+p \\ \rho uv \\ u(e+p) \end{bmatrix}, \quad \mathbf{g}(\mathbf{u}) = \begin{bmatrix} \rho u \\ \rho uv \\ \rho v^2+p \\ v(e+p) \end{bmatrix}. \tag{11b,c,d}$$

Here, u and v are the velocity components of the fluid in the x and y directions, ρ is the fluid density, e is the total energy of the fluid per unit volume, and p is the fluid pressure. For an ideal gas

$$p = (\gamma - 1)[e - \rho(u^2 + v^2)/2], \tag{11e}$$

where γ is the ratio of the specific heat at constant pressure to that at constant volume.

We solve a problem where a Mach 10 shock in air ($\gamma = 1.4$) moves down a channel containing a wedge with a half-angle of thirty degrees. This problem was used by Woodward and Collela [32] to compare several finite difference schemes on uniform grids. Like them, we orient a rectangular computational domain, $-0.3 \le x \le 3.4$, $0 \le y \le 1$, so that the top edge of the wedge is on the bottom of the domain in the interval $y = 0$, $1/6 \le x \le 3.4$. Thus, in the computational domain it appears like a Mach 10 shock is impinging on a flat plate at an angle of sixty degrees. The initial conditions that are appropriate for this situation are

$$\rho = 8.0, \quad p = 116.5, \quad e = 563.5, \quad u = 4.125\sqrt{3}, \quad v = -4.125,$$
$$\text{if } y < \sqrt{3}(x - 1/6), \tag{12a}$$

and

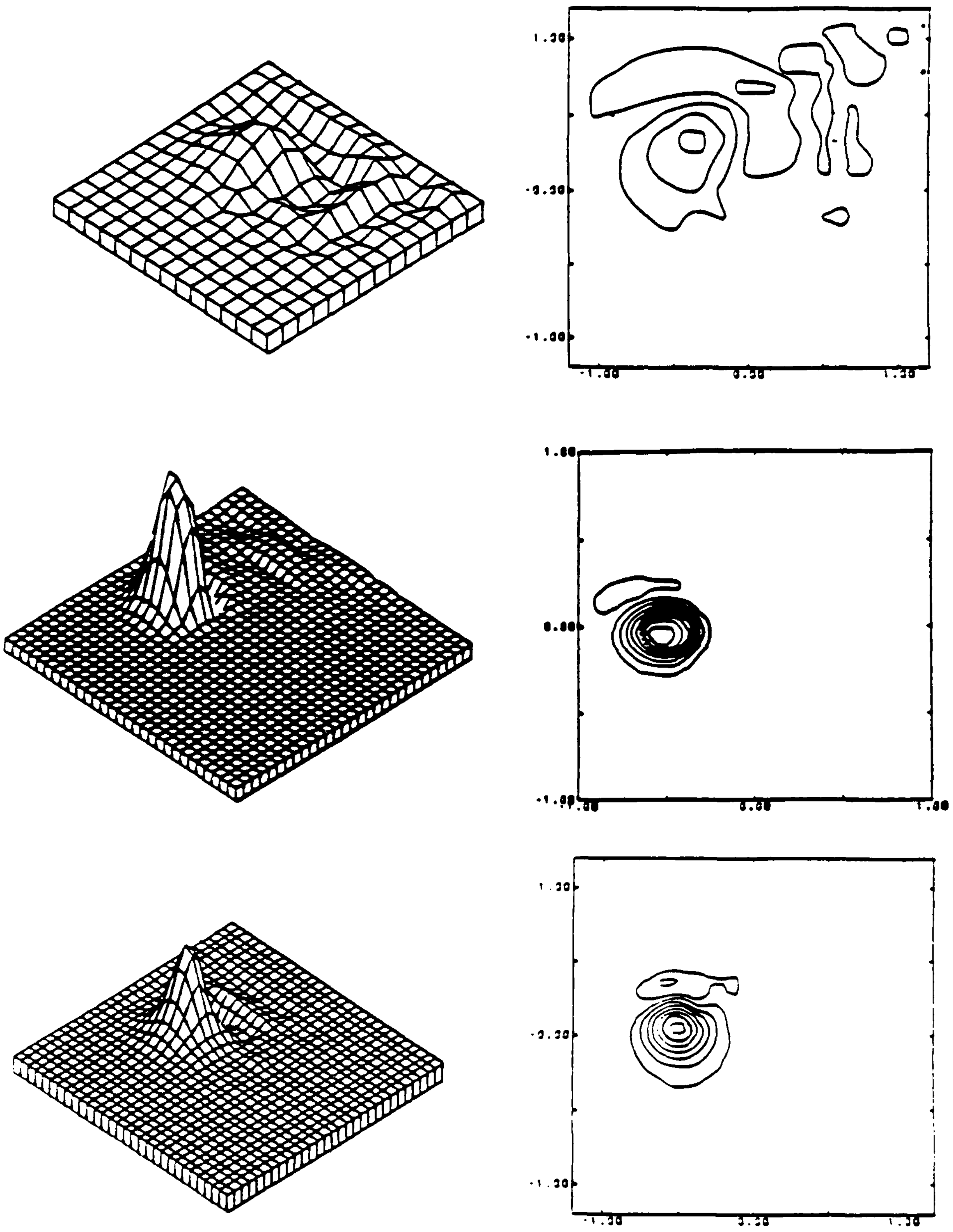

Figure 7. Surface and contour plots for Solutions 1, 2, and 3 (top to bottom) at t = 3.2 of Example 1.

$$\rho = 1.4, \quad p = 1.0, \quad e = 2.5, \quad u = 0, \quad v = 0, \qquad \text{if } y \geq \sqrt{3}(x-1/6). \tag{12b}$$

Along the left boundary ($x = -0.3$) and the bottom boundary to the left of the wedge ($y = 0$, $-0.3 \leq x \leq 1/6$), we prescribe Dirichlet boundary conditions according to Eq. (12); along the top boundary ($y = 1$), values are prescribed that describe the exact motion of an undisturbed Mach 10 shock; along the right boundary ($x = 3.4$), all normal derivatives are set to zero; and along the wedge ($y = 0$, $1/6 \leq x \leq 3.4$), reflecting boundary conditions are used.

The solution of this problem is a complete self-similar structure called a double-Mach reflection that was described in Ben-Dor and Glass [12, 13]. Two reflected Mach shocks form with their associated Mach stems and contact discontinuities. The geometry of these structures is very fine and is primarily confined to a small region that moves along the wedge with the incident shock. One of the two contact discontinuities is so weak that it is usually not noticed in computations.

The MacCormack finite difference scheme needs artificial viscosity to "capture" shocks without excessive oscillations. We used a model developed by Davis [20] which is total variation diminishing in one-space dimension.

Five solutions of this problem were calculated for $0 < t \leq 1.9$ as indicated in Table 3. Refinement was restricted to a maximum of two levels and a tolerance of 0.6 in the maximum norm was prescribed. A pointwise error indicator based on the assumption of

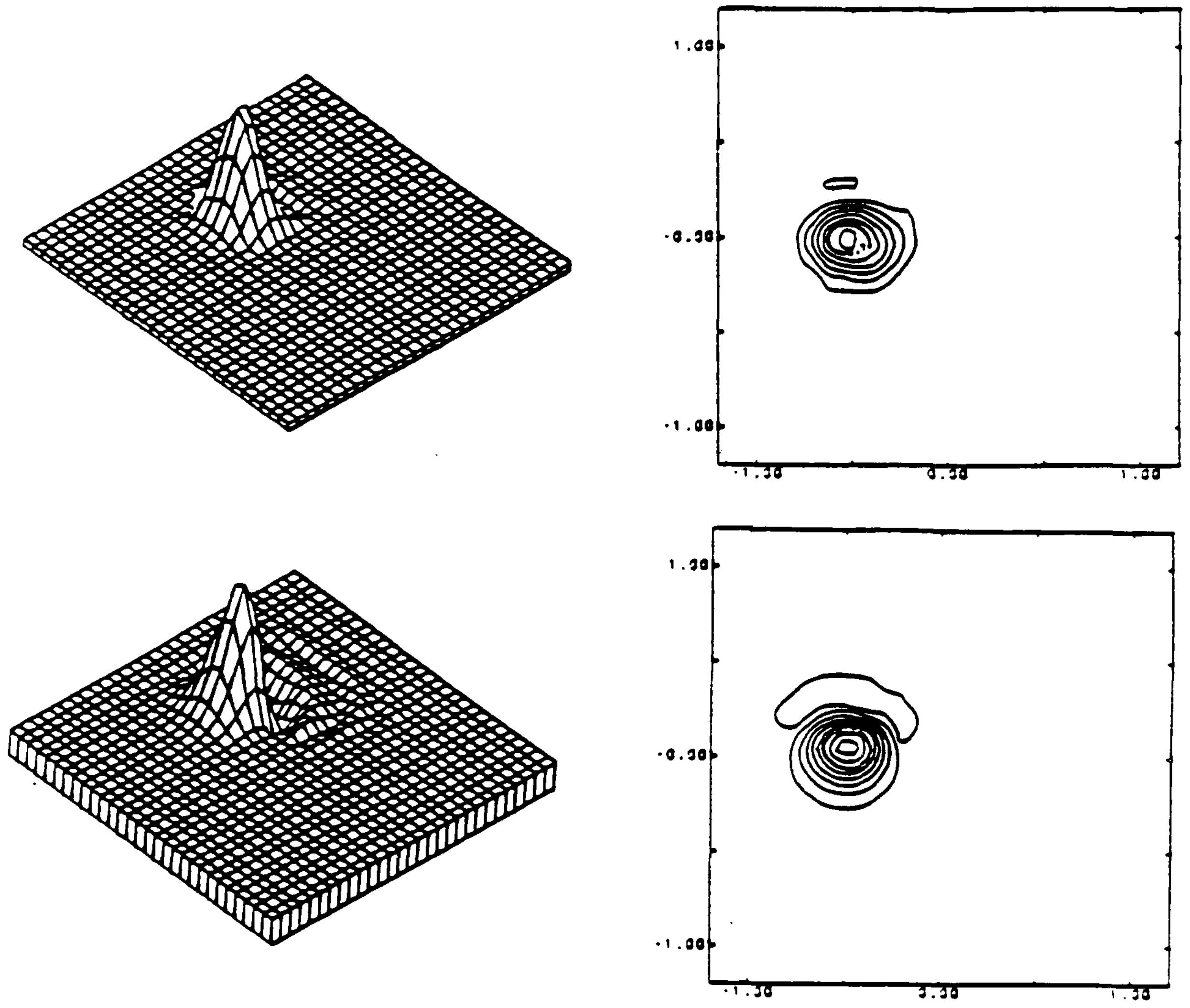

Figure 8. Surface and contour plots for Solutions 4 (top) and 5 (bottom) at $t = 3.2$ of Example 1.

Ref. No.	Strategy	Base Mesh	$\|e\|_1$	$\|e\|_\infty$	CPU Time (sec.)
1	Stationary *uniform mesh*	64×34	0.066	0.26	710
2	Moving mesh	44×20	0.056	0.18	340
3	Stationary mesh with refinement	44×20	0.055	0.23	719
4	Moving mesh with refinement	44×20	0.039	0.16	609

Table 2. Errors at $t = 1.2$ and computational costs for four solutions of Example 2.

smooth solutions, like the present one, is not appropriate for problems having discontinuities. Without restricting the maximum level of refinement, we could refine indefinitely in the vicinity of a discontinuity.

Solutions 2 through 5 were intended to be of comparable accuracy and we shall attempt to appraise the computational cost of each adaptive strategy. The maximum number of nodes that was introduced in any base time step and the total CPU time are presented as measures of computational complexity in Table 3. Contours of the density at $t = 0.19$ are shown for all five solutions in Figure 10 and the grids that were generated for Solution 4 at $t = 0.038, 0.076, 0.114, 0.152$, and 0.19 are shown in Figure 11.

As in the previous two examples, the mesh moving strategy of Solution 2 does a

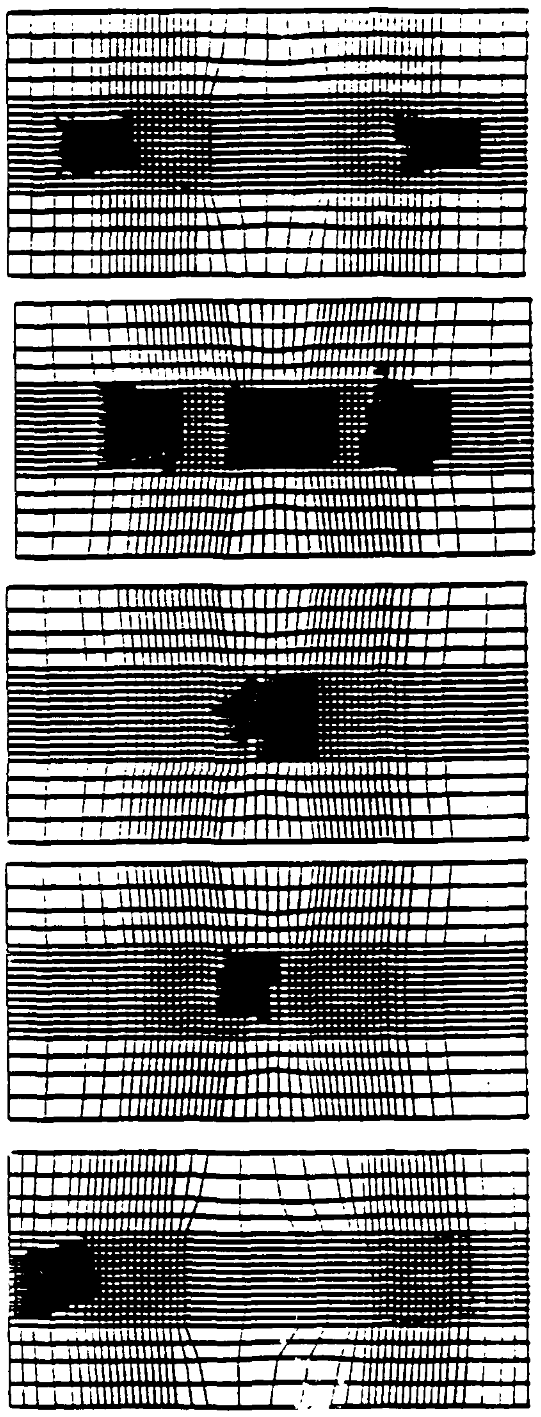

Figure 9. Grids created for Solution 4 of Example 2 at t = 0, 0.23, 0.46, 0.92, and 1.2 (top to bottom).

Ref. No.	Strategy	Base Mesh	Max. No. Nodes	CPU Time (sec.)
1	Stationary uniform mesh	63×29	1827	2130
2	Moving mesh	63×29	1827	2220
3	Stationary mesh with refinement	29×11	2782	3254
4	Moving mesh with refinement	29×11	3540	3725
5	Stationary uniform mesh	120×40	4800	6861

Table 3. Maximum number of nodes in any base time step and computational costs for five solutions of Example 3.

great deal to improve the results of the static Solution 1 for approximately a five-percent increase in computational cost. Comparing the top two contours oı Figure 10, we see that the resolution of the incident and reflected shocks is much finer with Solution 2 than with Solution 1. Additional detail of the structures in the Mach stem region and of the contact discontinuities is present in Solution 2, but not in the nonadaptive Solution 1. Finally, Solutions 1 and 5 display more oscillatory behavior behind the incident shock near the upper boundary. This is undoubtedly due to our maintaining a discontinuity where the shock intersects the upper boundary.

The use of refinement on a stationary mesh again does not give the dramatic improvement obtained by mesh moving (cf. the second and third contours of Figure 10).

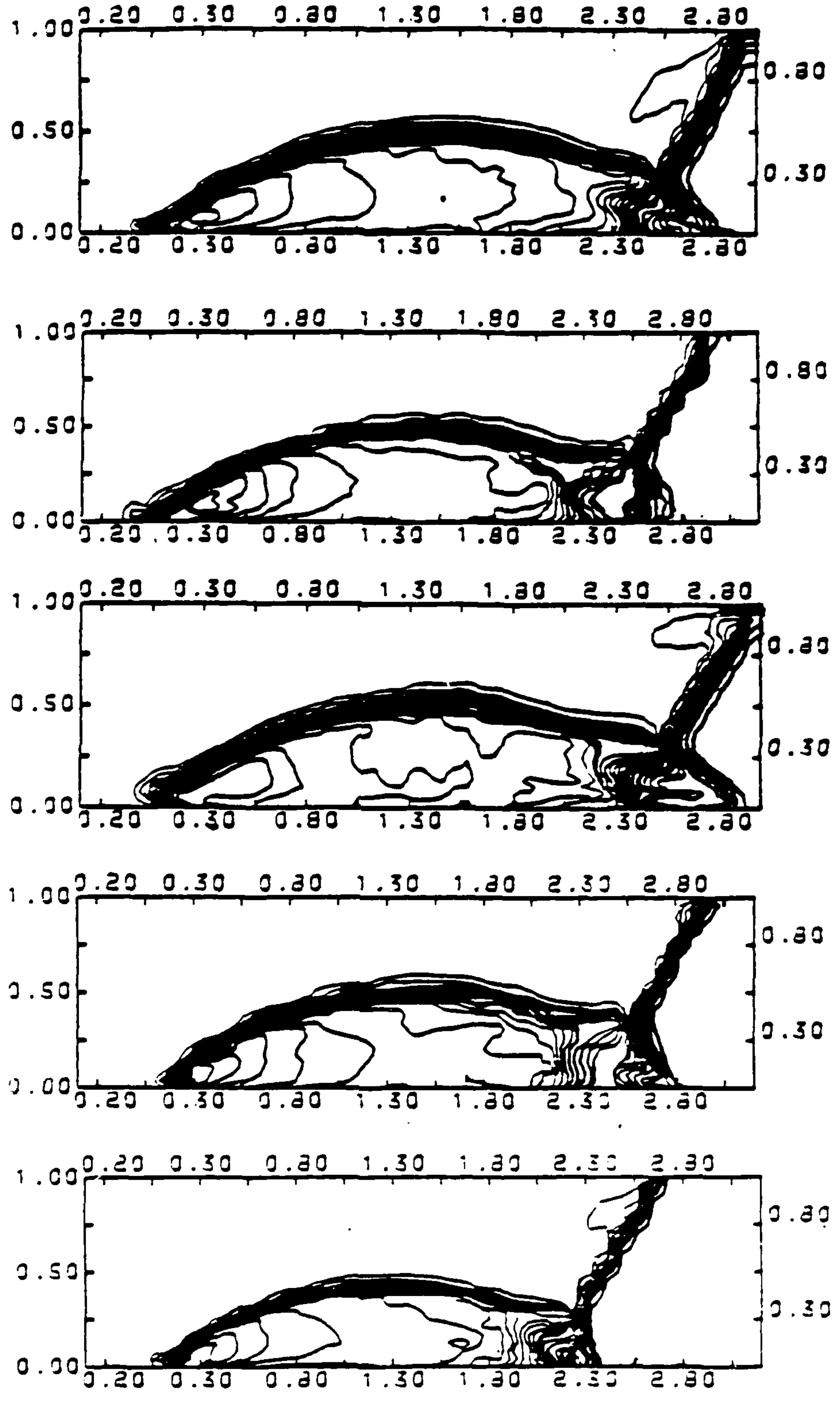

Figure 10. Contours of the density at t = 0.19 for Solutions 1 to 5 (top to bottom) of Example 3.

Initally the fine meshes were following the incident and reflected shock structures and better results were obtained; however, by $t = 0.19$ refinement is being performed over much of the domain and two levels of refinement are not sufficient for adequate resolution (cf. Arney and Flaherty [6]). The combination of mesh motion and refinement depicted by Solution 4 in Figure 10 provides a marked improvement in resolution. The sequence of meshes shown in Figure 11 shows that the coarse mesh is able to follow the differing dynamic structures and that refinement is only performed in the vicinity of discontinuities. Initially, only one rectangular cluster is needed to follow the incident shock (cf. Arney and Flaherty [5]). As time progresses, two clusters are created in order to follow the incident and reflected shocks (cf. the upper three meshes of Figure 11). A third cluster is created as time increases further in order to follow the evolving activity in the region of the Mach stem (cf. the lower two meshes of Figure 11).

Severe distortion of the mesh in the reflected shock region caused a static mesh regeneration to occur for Solution 4 at $t = 0.162$. The base meshes before and after the static regeneration are shown in Figure 12. Thus, Solution 4 demonstrates all of the capabilities of our adaptive procedure. The results presented in Table 3 and Figure 10 also show that Solution 4 provided greater resolution than the uniform mesh Solution 5 for approximately one-half of the cost. Solution 4 also shows many of the same characteristics as the solution computed by Woodward and Collela [32] using MacCormack's method on a 240×120 uniform grid. We were unable to compute a solution on such a fine mesh

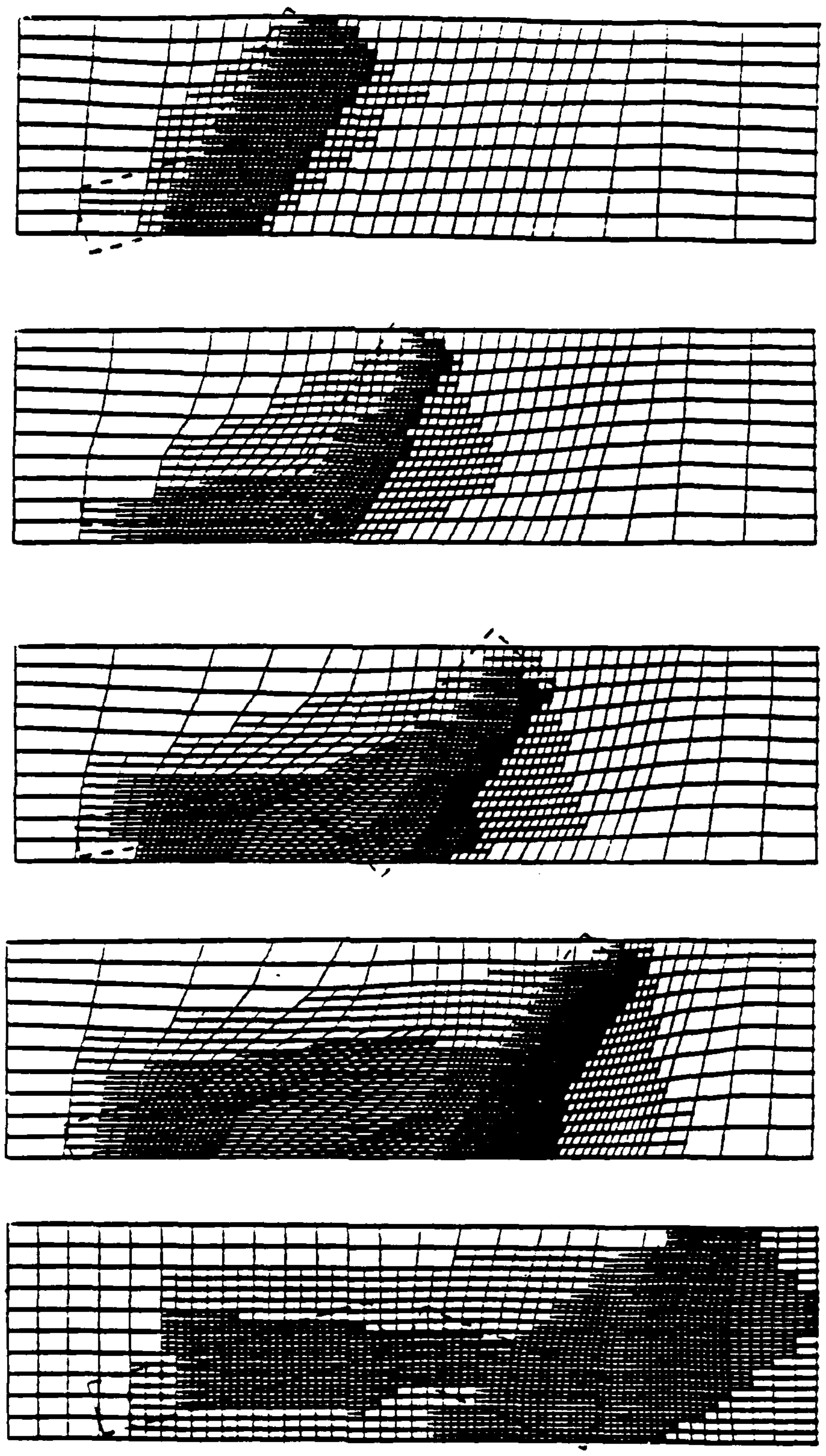

Figure 11. Grids created for Solution 4 of Example 3 at t = .038, 0.076, 0.114, 0.152, and 0.19 (top to bottom).

due to virtual memory limitations on our computer; however, we estimate that it would have used 14,400 nodes and 40,000 CPU seconds.

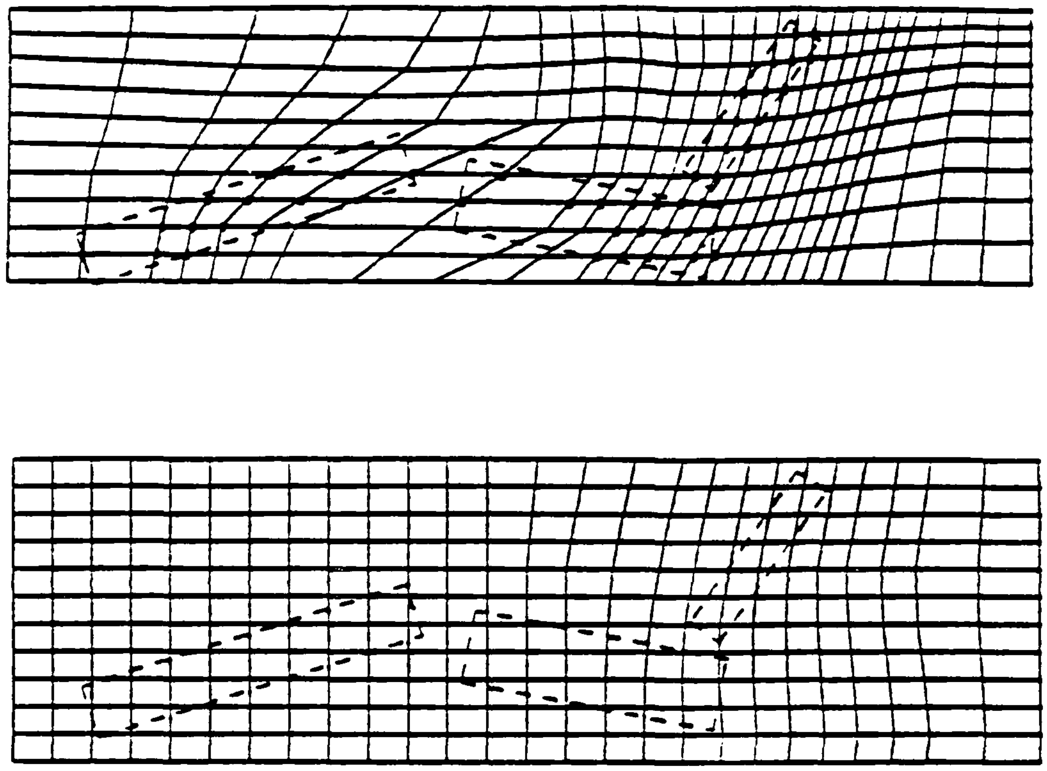

Figure 12. Base grids before (top) and after (bottom) the static mesh regeneration that was performed for Solution 4 of Example 3 at $t = 0.162$.

The results presented for this problem demonstrate the power and efficiency of our adaptive techniques; however, we would have preferred to allow more than two levels of refinement and a finer base mesh. These calculations would have produced better resolution of the discontinuities and other fine-scale structures that further demonstrate the computational advantages of adaptive methods relative to uniform mesh techniques. As noted, restrictions of our computing environment prevented us from doing this in a reasonable manner. We hope to perform these calculations in the future using a larger computing system.

IV. DISCUSSION OF RESULTS AND CONCLUSIONS.

We have described an adaptive procedure for solving systems of time-dependent partial differential equations in two-space dimensions that combines existing mesh moving [5] and local refinement [6] techniques. The algorithm also contains procedures for initial mesh generation and static mesh regeneration. It can be used with a wide variety of finite difference or finite element schemes and error indicators.

We obtained computational results for hyperbolic systems of conservation laws by using our adaptive methods with a MacCormack finite difference scheme and using Richardson's extrapolation to furnish local error indicators. Our computational results on three examples indicate that mesh moving can significantly reduce errors for approximately a ten-percent increase in cost relative to computations performed on stationary uniform meshes. The use of local refinement without mesh moving provided increased efficiency relative to uniform-mesh calculations, although not as dramatic as those found using mesh moving. The combination of mesh moving and local refinement provided reliable results while costing significantly less than stationary-mesh calculations. Thus, the overhead associated with the dynamic data structures is less than the time to calculate a comparable solution on a uniform mesh.

The results of Section III and others (cf. Arney and Flaherty [5, 6]) indicate that our mesh moving procedures perform better alone than with refinement. This is because the projection of fine-mesh solutions onto coarser meshes reduces the errors at base mesh nodes, and mesh motion based on controlling small or zero local discretization errors either fails or results in no movement. Erratic mesh motion can also occur with some techniques when movement indicators are small. This topic is discussed in Coyle et al. [19] and a possible remedy for one-dimensional problems is suggested in Adjerid and Flaherty [2]. Further experimentation and analysis are being performed in order to determine the best way to combine mesh moving and refinement.

There are several other ways to improve the efficiency, reliability, and robustness of our adaptive methods. The present Richardson's extrapolation-based error indicator is expensive and we are seeking ways of replacing it by techniques using p-refinement. Such methods have been shown [1, 2, 3, 8, 10, 16, 22] to have an excellent cost performance ratio when used in conjunction with finite element methods. An appropriate error indicator or estimator can be used to control a differential refinement algorithm, where different refinement factors (i.e., other than binary) are used in different high-error clusters. If the error indicator is capable of providing separate estimates of the spatial and temporal errors, as the present one does, then different refinement factors can also be used in space and time. We also hope to demonstrate the flexibility of our refinement procedure by using it with a finite difference or finite element scheme for parabolic problems.

The greater reliability and efficiency of adaptive techniques will be most beneficial in three dimensions. These techniques must be able to take advantage of the latest advances in vector and parallel computing hardware. The tree is a highly parallel structure and we have been developing solution procedures that exploit this in a variety of parallel computing environments.

REFERENCES

1. S. Adjerid and J. E. Flaherty, "A Moving Finite Element Method with Error Estimation and Refinement for One-Dimensional Time Dependent Partial Differential Equations," *SIAM J. Numer. Anal.*, 23 (1986), pp. 778-796.

2. S. Adjerid and J. E. Flaherty, "A Moving Mesh Finite Element Method with Local Refinement for Parabolic Partial Differential Equations," *Comp. Meths. Appl. Mech. Engr.*, 56 (1986), pp. 3-26.

3. S. Adjerid and J. E. Flaherty, "A Local Refinement Finite Element Method for Two-Dimensional Parabolic Systems," Tech. Rep. No. 86-7, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, 1986.

4. D.C. Arney, "*An Adaptive Mesh Algorithm for Solving Systems of Time-Dependent Partial Differential Equations*," Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, 1985.

5. D.C. Arney and J.E. Flaherty, "A Two-Dimensional Mesh Moving Technique for Time Dependent Partial Differential Equations," Tech. Rep. No. 85-9, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, 1985. Also *J. Comput. Phys.*, 67 (1986), pp. 124-144.

6. D.C. Arney and J.E. Flaherty, "An Adaptive Local Mesh Refinement Method for Time-Dependent Partial Differential Equations," Tech. Rep. No. 86-10, Department

of Computer Science, Rensselaer Polytechnic Institute, Troy, 1986.

7. I. Babuska, J. Chandra, and J.E. Flaherty, Eds., "*Adaptive Computational Methods for Partial Differential Equations*," SIAM, Philadelphia, 1983.

8. I. Babuska, A. Miller, and M. Vogelius, "Adaptive Methods and Error Estimation for Elliptic Problems of Structural Mechanics," in *Adaptive Computational Methods for Partial Differential Equations*, I. Babuska, J. Chandra, J. E. Flaherty, Eds., SIAM, Philadelphia, 1983, pp. 57-73.

9. I. Babuska, O.C. Zienkiewicz, J.R. Gago, and E.R. de A. Olivera, Eds., *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, John Wiley and Sons, Chichester, 1986.

10. R.E. Bank and A. Weiser, "Some A Posteriori Error Estimates for Elliptic Partial Differential Equations," *Maths. Comp.*, 44 (1985), pp. 283-301.

11. J.B. Bell and G.R. Shubin, "An Adaptive Grid Finite Difference Method for Conservation Laws," *J. Comput. Phys.*. 52 (1983), pp. 569-591.

12. G. Ben-Dor and I.I. Glass, "Non-Stationary Oblique Shock-Wave Reflections: Actual Isopycnics and Numerical Experiments," *AIAA J.*, 16 (1978), pp. 1146-1153.

13. G. Ben-Dor and I.I. Glass, "Domains and Boundaries of Non-Stationary Oblique Shock-Wave Reflections. 1. Diatomic Gas," *J. Fluid Mech.*, 92 (1979), pp. 459-496.

14. M. Berger, "On Conservation at Grid Interfaces," ICASE Rep. No. 84-43, ICASE, NASA Langley Research Center, Hampton, 1984.

15. M. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. Comput. Phys.*, 53 (1984), pp. 484-512.

16. M. Bieterman, J.E. Flaherty, and P.K. Moore, "Adaptive Refinement Methods for Non-Linear Parabolic Partial Differential Equations," Chap. 19 in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, I. Babuska, O.C. Zienkiewicz, J.R. Gago, and E.R. de A. Olivera, Eds., John Wiley and Sons, Chichester, 1986.

17. J.U. Brackbill and J.S. Saltzman, "Adaptive Zoning for Singular Problems in Two Dimensions," *J. Comput. Phys.*, 46 (1982), pp. 342-368.

18. S.R. Chakravarthy and S. Osher, "Computing with High-Resolution Upwind Schemes for Hyperbolic Equations," in *Large-Scale Computations in Fluid Mechanics*, B.E. Engquist, S. Osher, and R.C.J. Somerville, Eds., Lectures in Applied Mathematics, 22-1, AMS, Providence, 1985, pp. 57-86.

19. J.M. Coyle, J.E. Flaherty, and R. Ludwig, "On the Stability of Mesh Equidistribution Strategies for Time-Dependent Partial Differential Equations," *J. Comput. Phys.*, 62 (1986), pp. 26-39.

20 S. Davis, "TVD Finite Difference Schemes and Artificial Viscosity," ICASE Rep.

No. 84-20, NASA CR No. 172373, ICASE, NASA Langley Research Center, Hampton, 1984.

21. S.F. Davis and J.E. Flaherty, "An Adaptive Finite Element Method for Initial-Boundary Value Problems for Partial Differential Equations," *SIAM J. Sci. Stat. Comput.*, 3 (1982), pp. 6-27.

22. M. R. Dorr, "The Approximation Theory for the P-Version of the Finite Element Method," I, *SIAM J. Numer. Anal.*, 21 (1984), pp. 1180-1207.

23. H. A. Dwyer, "Grid Adaption for Problems with Separation, Cell Reynolds Number, Shock-Boundary Layer Interaction, and Accuracy," AIAA paper No. 83-0449, AIAA 21st Aerospace Sciences Meeting, Reno, 1983.

24. R.J. Gelinas, S.K. Doss, and K. Miller, "The Moving Finite Element Method: Applications to General Partial Differential Equations with Multiple Large Gradients," *J. Comput. Phys.*, 40 (1981), pp. 202-249.

25. A. Harten and J.M. Hyman, "Self-Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws," *J. Comput. Phys.*, 50 (1983), pp. 235-269.

26. R. Hindman, "Generalized Coordinate Forms of Governing Fluid Equations and Associated Geometrically Induced Errors," *AIAA J.*, 20 (1982), pp. 1359-1367.

27. R.W. MacCormack, "The Effect of Viscosity in Hypervelocity Impact Cratering," AIAA Paper 69-354, 1969.

28. G. McRae, W. Goodin, and J. Seinfeld, ''Numerical Solution of the Atmospheric Diffusion Equation for Chemically Reacting Flows,'' *J. Comput. Phys.*, 45 (1982), pp. 1-42.

29. M.M. Rai, ''Patched-Grid Calculations with the Euler and Navier-Stokes Equations,'' SIAM National Meeting, Boston, July 1986.

30. M.M. Rai and D. Anderson, ''Grid Evolution in Time Asymptotic Problems,'' *J. Comput. Phys.*, 43 (1981), pp. 327-344.

31. J.F. Thompson, Ed., *Numerical Grid Generation*, North-Holland, New York, 1982.

32. P. Woodward and P. Collela, ''The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks, *J. Comput. Phys.*, 54 (1984), pp. 115-173.

TECHNICAL REPORT INTERNAL DISTRIBUTION LIST

	NO. OF COPIES
CHIEF, DEVELOPMENT ENGINEERING BRANCH	
ATTN: SMCAR-CCB-D	1
-DA	1
-DC	1
-DM	1
-DP	1
-DR	1
-DS (SYSTEMS)	1
CHIEF, ENGINEERING SUPPORT BRANCH	
ATTN: SMCAR-CCB-S	1
-SE	1
CHIEF, RESEARCH BRANCH	
ATTN: SMCAR-CCB-R	2
-R (ELLEN FOGARTY)	1
-RA	1
-RM	1
-RP	1
-RT	1
TECHNICAL LIBRARY ATTN: SMCAR-CCB-TL	5
TECHNICAL PUBLICATIONS & EDITING UNIT ATTN: SMCAR-CCB-TL	2
DIRECTOR, OPERATIONS DIRECTORATE ATTN: SMCWV-OD	1
DIRECTOR, PROCUREMENT DIRECTORATE ATTN: SMCWV-PP	1
DIRECTOR, PRODUCT ASSURANCE DIRECTORATE ATTN: SMCWV-QA	1

NOTE: PLEASE NOTIFY DIRECTOR, BENET LABORATORIES, ATTN: SMCAR-CCB-TL, OF ANY ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST

	NO. OF COPIES
ASST SEC OF THE ARMY	
RESEARCH AND DEVELOPMENT	
ATTN: DEPT FOR SCI AND TECH	1
THE PENTAGON	
WASHINGTON, D.C. 20310-0103	
ADMINISTRATOR	
DEFENSE TECHNICAL INFO CENTER	
ATTN: DTIC-FDAC	12
CAMERON STATION	
ALEXANDRIA, VA 22304-6145	
COMMANDER	
US ARMY ARDEC	
ATTN: SMCAR-AEE	1
SMCAR-AES, BLDG. 321	1
SMCAR-AET-O, BLDG. 351N	1
SMCAR-CC	1
SMCAR-CCP-A	1
SMCAR-FSA	1
SMCAR-FSM-E	1
SMCAR-FSS-D, BLDG. 94	1
SMCAR-MSI (STINFO)	2
PICATINNY ARSENAL, NJ 07806-5000	
DIRECTOR	
US ARMY BALLISTIC RESEARCH LABORATORY	
ATTN: SLCBR-DD-T, BLDG. 305	1
ABERDEEN PROVING GROUND, MD 21005-5066	
DIRECTOR	
US ARMY MATERIEL SYSTEMS ANALYSIS ACTV	
ATTN: AMXSY-MP	1
ABERDEEN PROVING GROUND, MD 21005-5071	
COMMANDER	
HQ, AMCCOM	
ATTN: AMSMC-IMP-L	1
ROCK ISLAND, IL 61299-6000	

	NO. OF COPIES
COMMANDER	
ROCK ISLAND ARSENAL	
ATTN: SMCRI-ENM	1
ROCK ISLAND, IL 61299-5000	
DIRECTOR	
US ARMY INDUSTRIAL BASE ENGR ACTV	
ATTN: AMXIB-P	1
ROCK ISLAND, IL 61299-7260	
COMMANDER	
US ARMY TANK-AUTMV R&D COMMAND	
ATTN: AMSTA-DDL (TECH LIB)	1
WARREN, MI 48397-5000	
COMMANDER	
US MILITARY ACADEMY	1
ATTN: DEPARTMENT OF MECHANICS	
WEST POINT, NY 10996-1792	
US ARMY MISSILE COMMAND	
REDSTONE SCIENTIFIC INFO CTR	2
ATTN: DOCUMENTS SECT, BLDG. 4484	
REDSTONE ARSENAL, AL 35898-5241	
COMMANDER	
US ARMY FGN SCIENCE AND TECH CTR	
ATTN: DRXST-SD	1
220 7TH STREET, N.E.	
CHARLOTTESVILLE, VA 22901	
COMMANDER	
US ARMY LABCOM	
MATERIALS TECHNOLOGY LAB	
ATTN: SLCMT-IML (TECH LIB)	2
WATERTOWN, MA 02172-0001	

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, US ARMY AMCCOM, ATTN: BENET LABORATORIES, SMCAR-CCB-TL, WATERVLIET, NY 12189-4050, OF ANY ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST (CONT'D)

	NO. OF COPIES
COMMANDER US ARMY LABCOM, ISA ATTN: SLCIS-IM-TL 2800 POWDER MILL ROAD ADELPHI, MD 20783-1145	1
COMMANDER US ARMY RESEARCH OFFICE ATTN: CHIEF, IPO P.O. BOX 12211 RESEARCH TRIANGLE PARK, NC 27709-2211	1
DIRECTOR US NAVAL RESEARCH LAB ATTN: MATERIALS SCI & TECH DIVISION	1
CODE 26-27 (DOC LIB) WASHINGTON, D.C. 20375	1
COMMANDER AIR FORCE ARMAMENT LABORATORY ATTN: AFATL/MN EGLIN AFB, FL 32542-5434	1
COMMANDER AIR FORCE ARMAMENT LABORATORY ATTN: AFATL/MNF EGLIN AFB, FL 32542-5434	1
METALS AND CERAMICS INFO CTR BATTELLE COLUMBUS DIVISION 505 KING AVENUE COLUMBUS, OH 43201-2693	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, US ARMY AMCCOM, ATTN: BENET LABORATORIES, SMCAR-CCB-TL, WATERVLIET, NY 12189-4050, OF ANY ADDRESS CHANGES.